FAST PARALLEL ALGORITHM FOR DISCRETE FOURIER TRANSFORM IN MULTI-MESH NETWORK

Somen De,* Amit Datta,** Asit B. Bhattacharya,*** and Mallika De**

Abstract

In this paper we propose a parallel algorithm for computing the discrete Fourier transform (DFT) coefficients of n^2 points on a multi-mesh (MM) architecture [D. Das, M. De, and B.P. Sinha, A new network topology with multiple meshes, *IEEE Transactions on Computers*, *48*(5), 1999, 536–551.] having n^4 processing elements. Each processor in the MM architecture has the same number of neighbours, that is, four as in the case of two-dimensional (2D) torus yet, the time complexity of the proposed algorithm on this MM architecture is $O(n)$ which may be contrasted with $O(n^2)$ time for computing n^2 – point DFT coefficients on a 2D torus having the same nature of processors.

Key Words

2D mesh, multi-mesh, wrap-around connection, DFT, FFT, VLSI

1. Introduction

Fourier transform has a significant role in the computational developments of twentieth century and is extensively used in different branches like computer science, communication, speech transmission, coding theory, image processing and signal processing. It is a worthy accomplishment if it is possible to devise an algorithm which is an order of magnitude faster than any previous implementation. When the improvement is for a process that has many applications, then that accomplishment has a significant impact on scientists and practitioners.

The linear transformations, that is, computations of the form AB, where A is an $n \times n$ matrix and B is an n-vector, can be applied to the computation of discrete Fourier transform (DFT) [1], where A is a special type of symmetric matrix (discussed in Section 2) having some characteristics. The parallel computation of the linear transformation $A \times B = C$ takes $O(n)$ time for implementing in an $n \times n$ 2D mesh [2]. The authors Somen De *et al.* in the present paper initially propose a parallel implementation of the same using multi-mesh (MM) architecture [3], [4] having n^4 processors, all having degree 4. The processors in the MM network having n^2 meshes of size $n \times n$ are arranged in rows and columns of meshes of size $n \times n$. The time complexity of the algorithm is $O(n)$ for n^2 points, thereby reducing the time complexity by an $O(n)$. Finally, the algorithm is modified for incorporating the generation of the special $n^2 \times n^2$ matrix of Fourier transformation in the MM network in $O(n)$ element-by-element multiplication time. By this the number of I/O ports used and data input time has been reduced significantly.

The paper is organized as follows. Section 2 describes the DFT and the properties of the Fourier matrix. Section 3 defines fast Fourier transform (FFT). Section 4.1 describes the MM topology and its properties. Section 4.2 deals with mesh-related architectures and the time complexities of the algorithms of various numeric and non-numeric problems implemented on those architectures. The technique used for performing parallel implementation of linear transformation in MM is explained in Section 5. Section 6 deals with a flowchart and the algorithm for the parallel implementation of linear transformation, its time complexity and handling of higher order transformation matrix. In Section 7, the parallel algorithm for DFT and its time complexity has been dealt with an example. Section 8 deals with comparison of the time complexities of DFT computation algorithms in different architectures. Section 9 gives the experimental results of simulation program and we conclude in Section 10.

2. The Discrete Fourier Transforms

The Fourier transform of a continuous function $a(t)$ is given by:

$$A(f) = \int_{\infty}^{\infty} a(t) e^{2\pi f t} dt \qquad (1)$$

where $i = \sqrt{-1}$. The variable t is used to represent time, and f is used to represent frequency. The Fourier transform

* Department of Physics, Bijoy Krishna Girls' College, Howrah, Howrah-1, India; e-mail: de_somen@rediffmail.com
** Department of Engineering & Technological Studies, University of Kalyani, West Bengal, India; e-mail: amitdatta_wb@yahoo.co.in, demallika@yahoo.com
*** Department of Physics, University of Kalyani, West Bengal, India, e-mail: asit1951@yahoo.com
Recommended by Prof. Hamid Sarbazi-Azad
(DOI: 10.2316/Journal.211.2014.4.211-1012)

is used to convert a function of time into a function of frequency.

The above Fourier transform could be represented as the DFT which handles sample points of $a(t)$, namely a_0, a_1, …, a_{N-1}. The DFT is expressed as:

$$A_j = \sum_{0}^{N-1} a_k e^{2\pi i j k/N}, 0 \le j \le N-1 \quad (2)$$

Equation (2) can be rewritten as:

$$A_j = \sum_{0}^{N-1} a_k \, \omega^{jk}, 0 \le j \le N-1 \quad (3)$$

where $\omega = e^{2\pi i/N}$ is the primitive Nth root of unity in the complex plane or written more compactly as:

$$A = F_N \times a \quad (4)$$

where entries of Fourier matrix F_N are given by:

$$\{F_N\}_{jk} = \omega^{jk} \quad (5)$$

and $A = [A_0 A_1 \ldots A_{N-1}]^T, a = [a_0 a_1 \ldots a_{N-1}]^T$.

So, the problem of finding Fourier transform reduces to multiplying the matrix F_N of order N by a vector "a" of order N, which normally requires $O(N^2)$ operations.

A hint that the Fourier transform can be computed faster comes from observing that the evaluation points are not arbitrary but are in fact very special. They are N powers (sometimes called twiddle factors) ω^j, for $0 \le j \le N-1$. The two simple properties of these Nth roots of unity are if $N = 2n$, then (i) $-\omega^j = \omega^{j+n}$ and (ii) ω^2 is primitive nth root of unity.

3. Fast Fourier Transform

DFT of a vector of length N can be computed either directly from the definition or via a dense matrix–vector multiplication in quadratic time. But DFT [5], [6] can also be computed in $O(N \log N)$ time using FFT [7]–[9] which exploits the symmetry of the Fourier matrix F. The basic idea is to use properties of the nth roots of unity to relate the Fourier transform of a vector of size n to two Fourier transforms on vectors of size $n/2$.

$$F_n x = \begin{bmatrix} I_{n/2} & D_{n/2} \\ I_{n/2} & D_{n/2} \end{bmatrix} \begin{bmatrix} F_{n/2} & x_{even} \\ F_{n/2} & x_{odd} \end{bmatrix} \quad (6)$$

where x_{even} denotes the vector of size $n/2$ consisting of x_0, x_2, …, x_{n-2} and x_{odd} denotes the vector consisting of x_1, x_3, …, x_{n-1}, the matrix $I_{n/2}$ is the $n/2 \times n/2$ identity matrix and the matrix $D_{n/2}$ is $n/2 \times n/2$ diagonal matrix whose kth diagonal entry is ω^k. The radix 2 Cooley-Tukey FFT [10] uses a divide-and-conquer methodology to compute the DFT. It has been assumed that N is a power of 2.

4. Multi-mesh and Mesh-related Architectures

The k-ary n-cube has been the most popular multicomputer interconnection network due to its desirable properties such as each of implementation has recursive structure and ability to explain communication locality found in many parallel applications to reduce message latency.

A k-ary n-cube network has an n-dimensional grid structure with k-nodes (processors) in each dimension, such that every node is connected to two other nodes in each dimension by direct communication links.

A (k, n)-torus has n dimensions and $N = k^n$ nodes. Each node is uniquely identified by an n-tuple in radix k. A (k, n)-mesh is (k, n) torus with wraparound connection missing. The well-known binary hypercube is the 2-n mesh.

A linear array has k elements in one dimension. One extreme is a linear array; the other extreme is hypercube topology of dimension n which has two nodes along each dimension.

The n-dimensional cube is generalization of 2D mesh to n dimensions. Each node in the n-D mesh, with the exception of those in the periphery, is connected to $2n$ other nodes, two along each of the dimensions.

Mesh of tree and pyramid are two interconnection networks which use a combination of 2D meshes and tree structures. In 2D MM topology, n^2 2D meshes of size $n \times n$ are arranged in rows and columns and are interconnected with other meshes in row and column directions. Section 4.1 describes the MM network topology, to be used by the proposed algorithm. This section also refers different applications of MM topology. In Section 4.2 mesh and related architectures are discussed. Two important aspects of topological properties of these architectures such as diameter and bisection width, and time complexities of a few important numeric and non-numeric applications have been presented in a tabular form in Section 4.2.

4.1 The MM Network

In a MM network [3], [4], [11], [12] shown in Fig. 1, there are n^2 meshes of size $n \times n$ each, which themselves are again arranged in n rows and n columns so that there will be n^4 processors in the network. Each $n \times n$ mesh in this network is termed as a block. A processor on the MM can be identified by a four tuple, that is, $P(\alpha, \beta, x, y)$ where (α, β) is used for block address as the αth row and βth column, and (x, y) denotes the processor address as xth row and yth column within that block. Each processor $P(\alpha, \beta, x, y)$ is connected to $P(\alpha, \beta, x \pm 1, y \pm 1)$, if they exist, using bidirectional links referred as intra-block links. There exist, however, some additional bi-directional connections termed as inter-block links among the corners and boundary processors given by the following rule:

(i) **Horizontal inter-block links:**
$P(\alpha, \beta, x, 0)$ are connected to $P(\alpha, x, \beta, n-1)$ for $0 \le x, \alpha, \beta \le n-1$.
As a special case, when $\beta = x$, the link interconnects two processors within the same block.

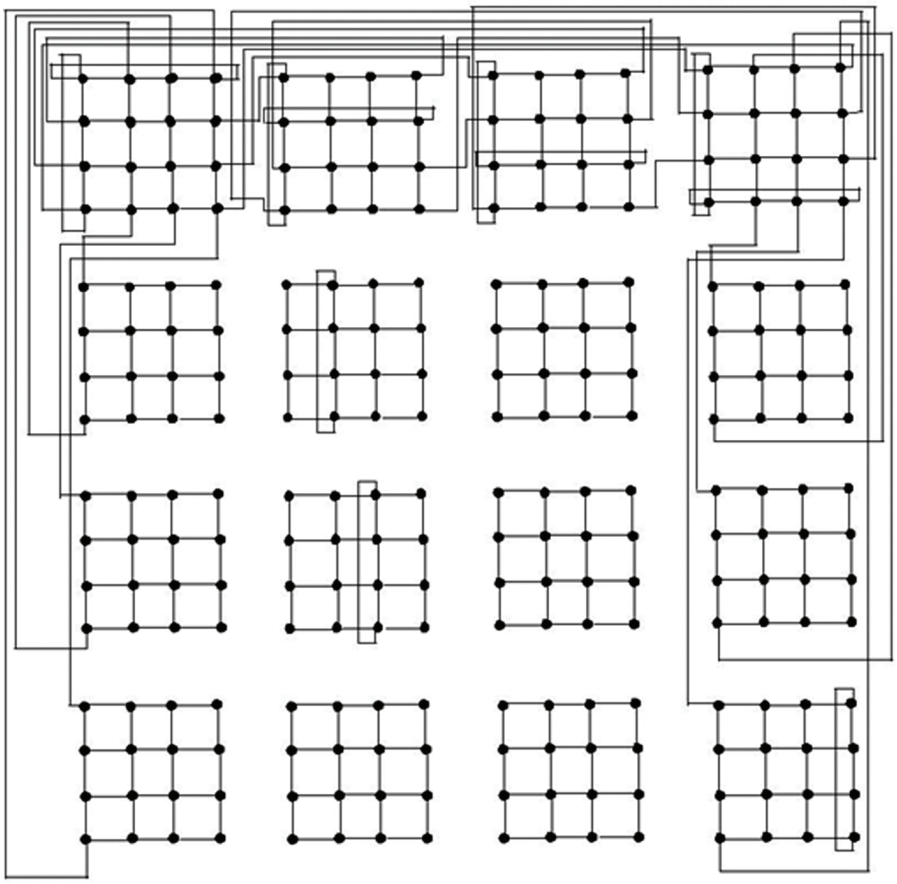


Figure 1. A simple $n \times n$ multi-mesh network with $n = 4$ (all links are not shown).

(ii) **Vertical inter-block links**:

$P(\alpha, \beta, 0, y)$ are connected to $P(y, \beta, n-1, \alpha)$ for $0 \leq y, \alpha, \beta \leq n-1$.

As a special case, when $\alpha = y$, the link interconnects two processors within the same block.

Rule (i) interconnects two blocks in the horizontal direction, whereas rule (ii) defines the vertical inter-block connections.

In a simple $n \times n$ mesh only $(n-2)^2$ internal processors have degree 4, the four corner processors are of degree 2 and $4(n-2)$ boundary processors have degree 3, as opposed to degree 4 for all processors on the MM. Moreover, the diameter of the network is $2n$ as opposed to $2(n^2-1)$ for an $n^2 \times n^2$ mesh. For this reason, any real-life applications can be solved on the proposed network more efficiently than on the corresponding mesh with the same number of processors. When time complexity is governed by the diameter of the network, the MM network is more advantageous than mesh.

As examples of real-life applications, simple problems like those of calculating the sum, average, minimum and maximum of n^4 data values have been implemented in $O(n)$ time on the MM network having n^4 processors [3]. Numeric problems like matrix multiplication and Lagrange's interpolation have been implemented in MM having n^4 processors in $O(n)$ and $O(\mathrm{p}^{0.6})$ time for n^2 point Lagrange's interpolation and $p \times p$ matrix multiplication, respectively, where $p = n^{5/3}$. Non-trivial problem like sorting of n^4 data values has also been implemented in $O(n)$ time [13]. In case of simple $n^2 \times n^2$ mesh, each of these problems takes $O(n^2)$ time. The reduced time complexity has been achieved due to the inter-block links among the boundary processors of the meshes as defined by the above two rules. Optical technology has been used in implementing MM network in [14], [15]. The inter-block communication is done in MM network through optical technology using wavelength division multiplexing in [15]. Sen *et al.* [16] introduced a new metric, *flow number*, for a generalized multi-mesh (GM) that can be used to evaluate topologies for optical networks. The design and evaluation of a highly scalable, decentralized and self-organizing peer-to-peer network based on MM topology has been presented in [17].

4.2 Related Architectures and Results

The difference between MM (having $N = k^4$ elements) and n-dimensional mesh is that MM is 2D where k^2 meshes of size $k \times k$ are themselves arranged in rows and columns, whereas in n-dimensional mesh, meshes of size $k \times k$ are arranged in n dimensions (having k^n elements), which gives an n-dimensional structure. Two important aspects of network topology, like diameter and bisection width are different for them. Diameters and bisection width of MM are $2k$, $k^3/2$ whereas for n-dimensional meshes those are $n(k-1)$ and k^{n-1}, respectively. The node degree is also different for the two architectures. For MM it is regular 4 degree, whereas for n-dimensional mesh it is $2n$ for internal

Table 1
Different Architectures with Time Complexities for Parallel Algorithms

	Diameter	Bisection Width	Prefix Sum	Sorting	Max/Min/ Average	Matrix Operations	Lagrange's Interpolation
Mesh $N = n \times n$	$2(\sqrt{N}-1)$	$\sqrt{N}$	$O(\sqrt{N})$	$3\sqrt{N}+O(\sqrt{N})$ [18]	$\sqrt{N}$	Matrix–vector product θ $(\sqrt{N})$ Matrix–matrix product $O(\sqrt{N})$ [19]	$10(\sqrt{N}-1)+O(1)$ No. of terms in the series $N+1$ [20]
Hypercube $(N=2^n)$	$\log N$	$N/2$	$\log N$	$O(n \log n/p)+$ $O(n)$ using p processors [21]		Matrix–matrix product $O(p)$ using p^2 processors $O(\log p)$ using p^3 processors $p^3 < N$ [22]	$\log N$ [23]
2D Mesh of tree $N = n \times n$ Total number of processors $= 3n^2-2n$ [24]	$4 \log \sqrt{N}$ [25]	$2\sqrt{N}$	$4 \log \sqrt{N}+$ $O(1)$ using n^2 processors	$\Omega(N^{1/2})$	O $(\log \sqrt{N})$	Matrix-vector product $O(\sqrt{N})$ Matrix–matrix product O $(\sqrt{N})$	$\log \sqrt{N}$ using N processors in base $\sqrt{N}$ is the no. of input data [26]
Pyramid of base $N = n \times n$ [27], [28] Total number of processors $= 4n^2/3-1/3$	$2 \log_4 N$	$2\sqrt{N}-2$	$\Omega(\log N)$	$\Omega(N^{1/2})$	$O(\log N)$		
Multi-dimensional mesh [28] $N = n^d$	$d(N^{1/d}-1)$	$N^{1-1/d}$	$\theta(N^{1/d})$	$\theta(N^{1/d})$	$\theta(N^{1/d})$	Matrix–matrix product $O(\log N)$ using N processors [29]	
MM $N = n^4$ [3]	$2N^{1/4}$	$N^{1/4}/2$	$O(N^{1/4})$	$O(N^{1/4})$ [13]	$O(N^{1/4})$	$(\sqrt{N} \times \sqrt{N})$ Matrix–vector product $O(N^{1/4})$ (proposed) $(p \times p)$ Matrix-by-matrix product $O(p^{0.6})$, where $p = n^{5/3}$	$O(N^{1/4})$

nodes, n for corner nodes, $n+1$ or $n+2$ for boundary nodes other than corners.

As further comparison with other architectures such as hypercube, mesh-of-tree and pyramid, the diameter, bisection width and time complexities of different algorithms which have already been implemented in MM topology have been mentioned in Table 1. It is evident from table that most of the implementations in MM have better time complexities than other architectures. In some implementations $\log N$ order complexities have been

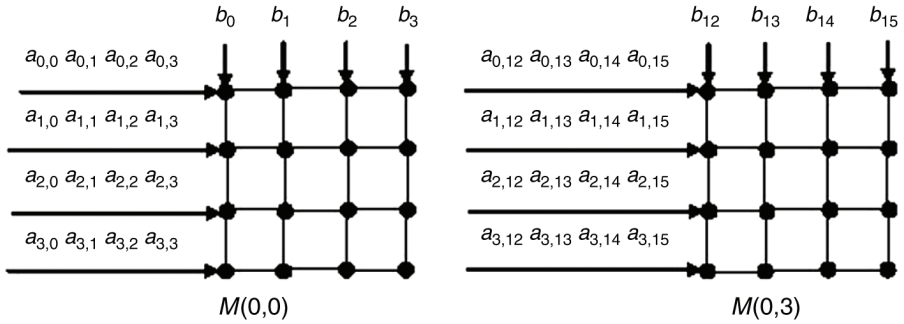


Figure 2. Data inputs along rows and columns of block $M(0, 0)$ and $M(0, 3)$ for $n = 4$.

achieved for hypercube and pyramid as they have higher total number of processors and have a tree structure above their base.

5. Linear Transformation by Matrix–Vector Multiplication

Linear transformation of a matrix A of size $p \times p$ by a vector B of size p is given by $C = A \times B$, where C is the transformed vector of length p.

5.1 Parallel Implementation using MM Network

Let us partition the matrix A as $n \times n$ sub-matrices $A_{\alpha,\beta}$, $0 \le \alpha, \beta \le n-1$, when $n = \sqrt{p}$. These n^2 sub-matrices are initially placed in the blocks $M(\alpha, \beta)$, $0 \le \alpha, \beta \le n-1$ of the MM network having n^4 processors, where each block contains the processors $P(\alpha, \beta, *, *)$. Here "*" indicate all possible values from 0 to $n-1$. The sub-vectors $A_{\alpha,\beta}$'s are placed in the blocks $M(\alpha, \beta)$'s, $0 \le \alpha, \beta \le n-1$, through the left boundaries.

Similarly, the vector B is partitioned into n column sub-vectors B_β, $0 \le \beta \le n-1$. The sub-vectors B_β's are placed in the processor blocks $M(*, \beta)$'s, $0 \le \beta \le n-1$. The inputs of sub-vectors B_β's are through the first rows of each block and are propagated down to the other rows of the same block as shown in Fig. 2.

Now each $M(\alpha, \beta)$ will compute $A_{\alpha,\beta} \times B_\beta$, $0 \le \alpha, \beta \le n-1$. An example showing the contents of $M(1, 2)$ is shown in Fig. 3.

To get $C_i = \sum_{j=0}^{n^2-1} p_{ij}$ for each i, where $p_{ij} = a_{ij} \times b_j$, all p_{ij}'s are to be brought in a single block $M(i/n, i\%n)$,

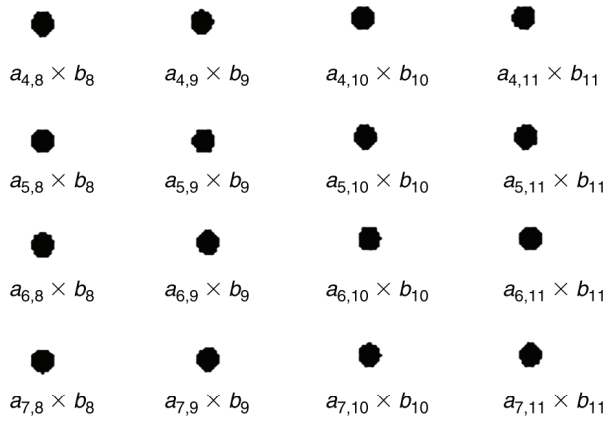


Figure 3. Partial products in the block $M(1, 2)$.

as they are now scattered in ith rows of n different blocks. This can be done using n shifts along the horizontal inter-block links of the MM network [as explained in Fig. 4(a) and (b)]. These n^2 components of each mesh are summed together to give an element of vector C [Fig. 4(c)] and are placed in the upper right corner of block $M(i/n, i\%n)$ [Fig. 4(d)]. One more single step of data movement along horizontal inter-block links bring them to the left boundary of all the blocks $M(\alpha, 0)$, $0 \le \alpha \le n-1$.

6. Parallel Implementation of Linear Transformation

In Section 6.1 a flowchart showing the detailed steps for parallel implementation of linear transformation is given. Section 6.2 deals with the parallel implementation of linear transformation algorithm (PLT) and Section 6.3 gives the time complexity of the parallel algorithm.

6.1 Flowchart of the Algorithm PLT

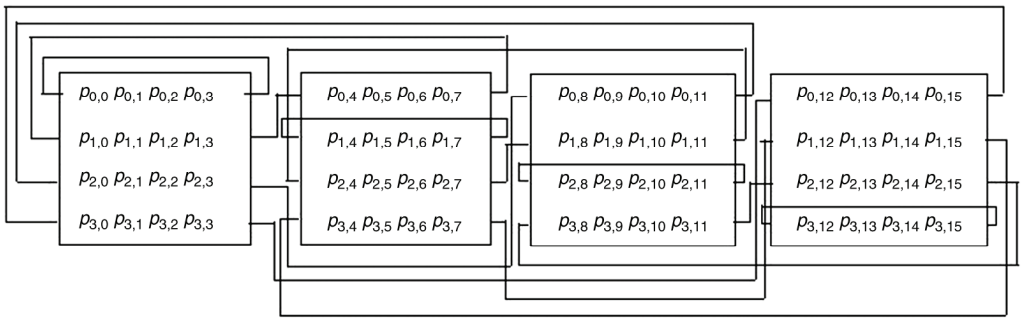
Figure 5 shows a flowchart indicating different steps of the parallel algorithm for linear transformation. The algorithm PLT of Section 6.2 is written following this flowchart.

6.2 Algorithm PLT

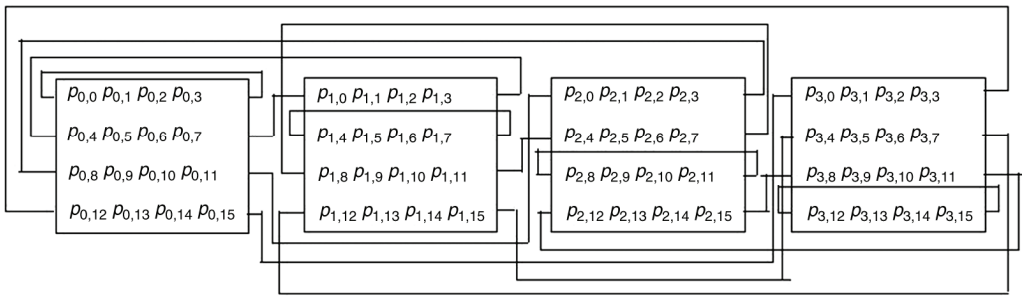
6.2.1 Initialization Step

The step is subdivided into two parts. Register H1s contain the initial values of matrix A, whereas registers H2s contain the initial values of vector B.

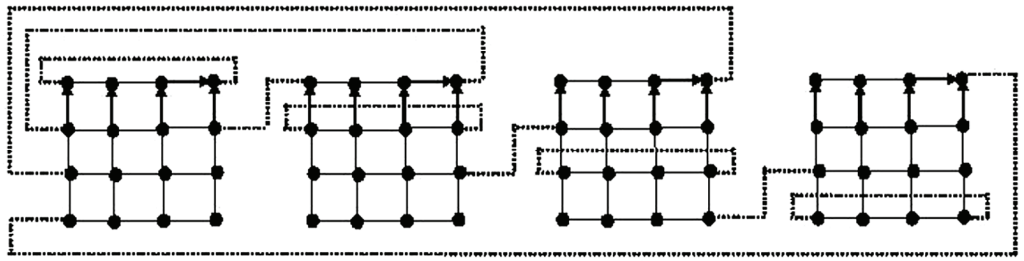
Step 1: $\forall\ \alpha, \beta$ and x, $0 \le \alpha, \beta, x \le n-1$ do in parallel
  1.1 H1$(\alpha, \beta, x, 0) \leftarrow a_{\alpha n+x, \beta n+(n-1)}$
  1.2 for $i = 1$ to $n-1$ do
    begin
      for $j = 1$ to i do in parallel
        H1 $(\alpha, \beta, x, i-j+1) \leftarrow$ H1$(\alpha, \beta, x, i-j)$;
        if $(j = i)$
          H1$(\alpha, \beta, x, 0) \leftarrow a_{\alpha n+x, \beta n+(n-1)-i}$;
      endfor
    end
Step 2: $\forall\ \alpha, \beta$ and y, $0 \le \alpha, \beta, y \le n-1$ do in parallel
  1.1 H2$(\alpha, \beta, 0, y) \leftarrow b_{\beta n+y}$;
  1.2 for $i = 1$ to $n-1$ do
      H2 $(\alpha, \beta, i, y) \leftarrow$ H2 $(\alpha, \beta, i-1, y)$;



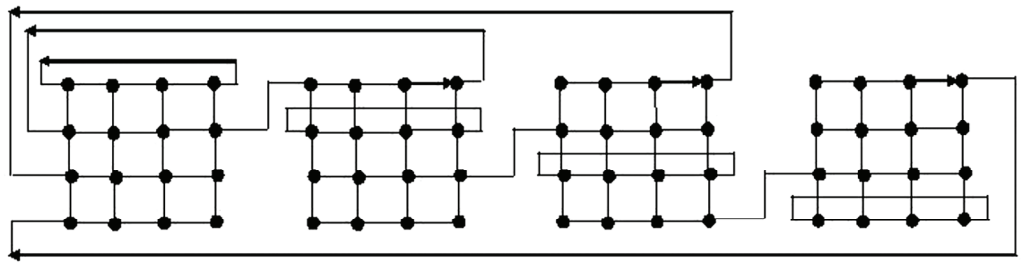
(a)



(b)



(c)



(d)

Figure 4. In the above series of images, (a) blocks $M(0, *)$'s contain the partial products in MM for $n = 4$; (b) contents of blocks $M(0, *)$'s after n steps of data movements along the horizontal inter-block links in MM for $n = 4$; (c) column sum and then 0th row sum in each block in MM for $n = 4$; and (d) single step data movement along horizontal inter-block links (solid lines with arrowhead indicate the direction of data movements).

6.2.2 Multiplication Step

$\forall\ \alpha, \beta, x$ and y, $0 \le \alpha, \beta, x, y \le n-1$ do in parallel
$\mathrm{H1}(\alpha, \beta, x, y) \leftarrow \mathrm{H1}(\alpha, \beta, x, y) \times \mathrm{H2}(\alpha, \beta, x, y)$;

6.2.3 Data Movement Step

In the MM network, horizontal inter-block link forms a cycle of length $2n$ between the kth row of the block $M(i, j)$ and the jth row of the block $M(i, k)$ for $j \neq k$, $0 \le i \le n-1$. For a given α, if we shift the data elements in $M(\alpha, *)$ through n positions along the horizontal cycles, then the ith row elements of $M(\alpha, j)$ will move to the jth row of $B(\alpha, i)$, $0 \le \alpha \le n-1$.

/* Here, '*' indicates all possible values from 0 to $n-1$, but the same value for it must be used on both sides of the assignment operator */

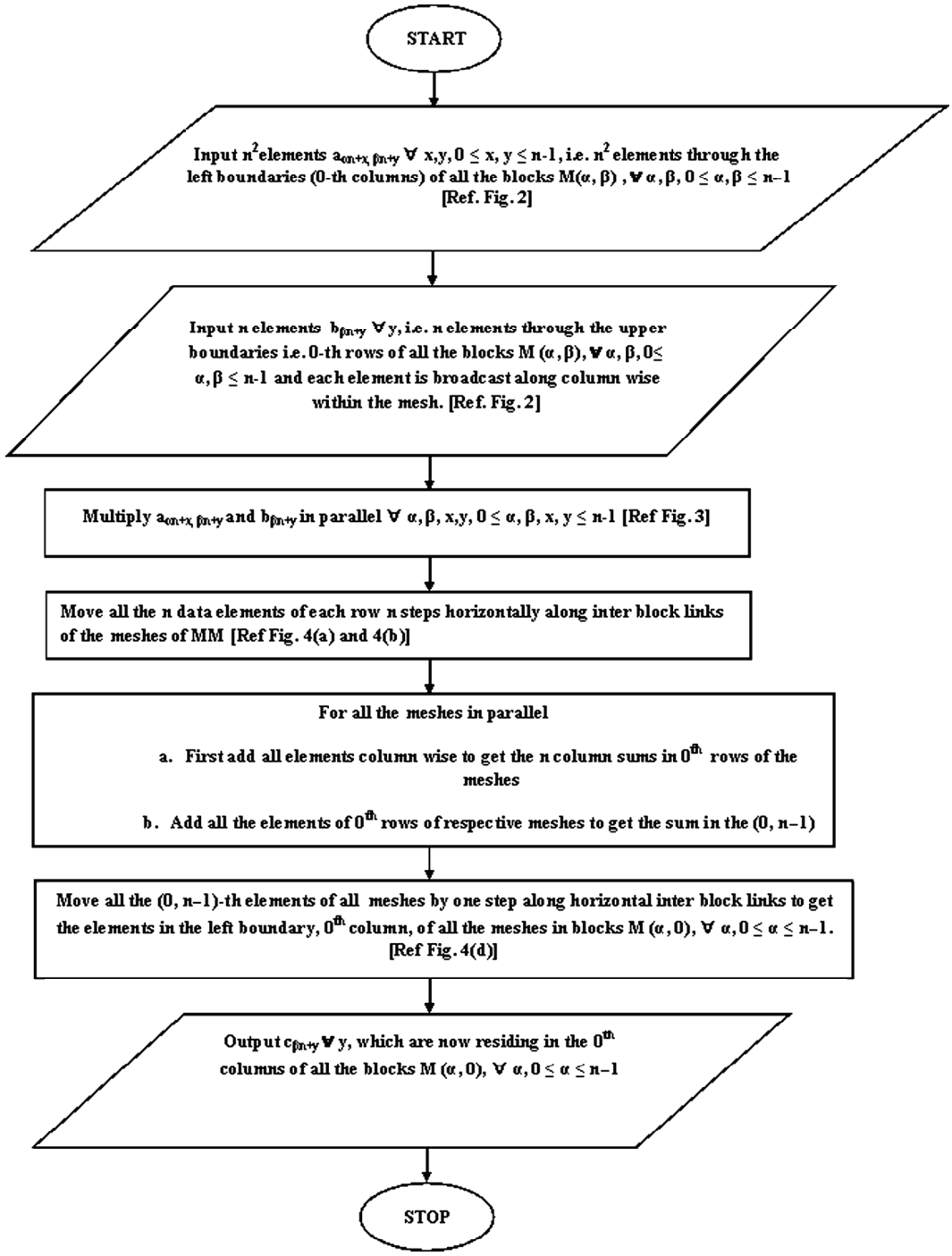


Figure 5. Flow chart for algorithm PLT.

∀ α and β, $0 \leq \alpha, \beta \leq n-1$ do in parallel
begin
H1(α, β,*, $n-1$) ← H1(α,*, β, 0);
for $j = 1$ downto $n-1$
if ($j = n-1$)
H1(α, β,*, $j-1$) ← H1(α, β, *, $n-1$);
else
H1(α, β,*, $j-1$) ← H1(α, β, *, j);
endfor
end

6.2.4 Addition Step

∀ α, β and y, $0 \leq \alpha, \beta, y \leq n-1$ do in parallel
begin
for $i = n-1$ downto 1 do
H1 (α, β, $i-1$, y) ← H1 (α, β, $i-1$, y) + H1 (α, β, i, y);
/* P(α, β, 0, y) contain the column sums */
for $j = 0$ to $n-2$ do
H1 (α, β, 0, $j+1$) ← H1 (α, β, 0, $j+1$) + H1 (α, β, 0, j);
/* Summing along the 0th row in each block, the sum of the n^2 data values of the block is finally brought to the processor P (α, β, 0, $n-1$) */
end

6.2.5 Data Arrangement Step

In this step the output data vector C is moved to the 0th column of all the blocks $B(\alpha, 0)$, $0 \le \alpha \le n-1$.

$\forall\ \alpha$ and β, $0 \le \alpha, \beta \le n-1$ do in parallel
H1 $(\alpha, 0, \beta, 0) \leftarrow$ H1 $(\alpha, \beta, 0, n-1)$;
/* single step horizontal data movement along inter-block links */

6.3 Time Complexity of Algorithm PLT

Steps 1 and 2 of initialization require n steps each. Multiplication step requires single step of element by element multiplication time. Data movement along horizontal inter-block links takes n steps. $2(n-1)$ data movement and addition steps are required for step D to add n^2 data elements of each mesh. Step E for output data arrangement requires single step. So, $5n-1$ data movement steps, 1 multiplication step and $2(n-1)$ addition steps are required for parallel implementation in a MM of n^4 processors resulting in an $O(n)$ algorithm for matrix–vector product. The AT^2 value is $O(n^6)$.

6.4 Multiplication of Higher Order Matrices

A square matrix A of size $kp \times kp$ can also be multiplied by a column vector B of size $kp \times 1$ using an MM network with only n^4 processors. Matrix A can be broken down into k^2 $\sqrt{p} \times \sqrt{p}$ components ($n = \sqrt{p}$) as A_{00}, A_{01}, A_{02}, …, $A_{0,k\sqrt{p}-1}$, A_{10}, A_{11}, A_{12}, …, $A_{1,k\sqrt{p}-1}$, …, $A_{k\sqrt{p}-1,0}$, $A_{k\sqrt{p}-1,1}$, …, $A_{k\sqrt{p}-1,k\sqrt{p}-1}$. Similarly, B can be broken down into k components as B_0, B_1, …, $B_{k\sqrt{p}-1}$. Output vector also comes as components C_0, C_1, …, $C_{k\sqrt{p}-1}$. Each block $M(i, j)$, $0 \le i, j \le n-1$, will now compute k^2 matrix–vector multiplication of sizes $n \times n$ and $n \times 1$. In general, we can say that the block $M(i, j)$ of the MM will compute the matrix–vector products $A_{(i+sn),(j+tn)}B_{(j+tn)}$, for $s = 0, 1, \ldots, k-1$ and $t = 0, 1, \ldots, k-1$. (As if, the submatrix components have been folded k-times in horizontal and vertical directions). For example, with $n = 4$ and $k = 3$, the block $M(0, 0)$ will now compute the components ($A_{00}B_0$, $A_{04}B_4$, $A_{08}B_8$), ($A_{40}B_0$, $A_{44}B_4$, $A_{48}B_8$) and ($A_{80}B_0$, $A_{84}B_4$, $A_{88}B_8$). Among the k^2 products elements to be computed at each processors, k are of same row and can be added by $(k-1)$ addition times. There is k such pairs. So, in general, $k(k-1)$ addition times reduce the elements at each processor to k for k different rows. Now, horizontal data movements along horizontal inter-block links, in general, require $k + (n-1)$ steps of data movements. Now, each block contains elements of k rows. Addition requires $k(n-1+k-1)$, that is, $k(n+k-2)$ steps to get k sums for k rows in $M(*,*, 0, 0)$ positions. From there, they can be brought to the 0th column of each block in 0th column of blocks (*i.e.*, having β value equal to 0) in $(k+n)$ steps.

So, overall time complexity will be $O(kn)$ and the AT^2 value $O(k^2n^6)$.

7. Parallel Implementation of DFT using MM Network

7.1 Algorithm PDFT

The elements $a_{\alpha n+i,\beta n+j}$, $0 \le \alpha, \beta, i, j \le n-1$, of the Fourier matrix can be obtained by multiplying the element $a_{\alpha,\beta}$, $0 \le \alpha, \beta \le n-1$, by $(\omega^\alpha)^n$ in row direction and $(\omega^{\beta+k})^n$ (or $(\omega^\beta)^n$ as $\omega^{kn} = 1$) in column direction where $k = 0, n, 2n, \ldots$. Initially, $a_{\alpha,\beta} = (\omega)^{\alpha\times\beta}$ are given as input to the (0,0) processor of blocks $M(\alpha, \beta)$, $0 \le \alpha, \beta \le n-1$ and then first row elements of blocks $M(\alpha, \beta)$, $0 \le \alpha, \beta \le n-1$ are computed by repeated multiplication of $a_{\alpha,\beta}$ by the row multiplier $(\omega^\alpha)^n$. Then all the first row elements are repeatedly multiplied by the column multiplier $(\omega^\beta)^n$ to generate the other rows of the blocks $M(\alpha, \beta)$. During the row generation process of Fourier matrix, the proper components of B vector are input through the 0th row of each block and propagated along vertical direction to the other rows of the same block. B is the vector which is to be transformed using the parallel implementation of discrete Fourier transform algorithm (PDFT). The above steps are done in parallel for all the blocks.

7.1.1 Initialization Step

The step is subdivided into four parts. Two local registers H1 and H2 are associated with each of the processing elements.

In this section the Fourier matrix A is generated in H1. And matrix B is input in H2.

$\forall\ \alpha, \beta$, $0 \le \alpha, \beta \le n-1$ do in parallel
Step 1:
1.1 H2 $(\alpha, \beta, 0, 0) \leftarrow (\omega^\alpha)^n$; /*Input the multiplier for row direction in H2*/
1.2 do in parallel
H1$(\alpha, \beta, 0, 0) \leftarrow \omega^{\alpha\times\beta}$;
H2 $(\alpha, \beta, 0, 1) \leftarrow$ H2 $(\alpha, \beta, 0, 0)$;
/* the n^2 values of $\omega^{\alpha\times\beta}$ are pre-calculated and placed at registers H1 at the upper left corner processor of each block M(α, β), $0 \le \alpha, \beta \le n-1$ */
Step 2: for $k = 1$ to $n-1$ do steps 2.1, 2.2 and 2.3 in parallel
2.1 H1 $(\alpha, \beta, 0, 1) \leftarrow$ H1 $(\alpha, \beta, 0, 0)$;
2.2 If $(k < n-1)$ then
H2 $(\alpha, \beta, 0, k+1) \leftarrow$ H2 $(\alpha, \beta, 0, k)$;
2.3 if $(k-1 = 0)$ then H2 $(\alpha, \beta, 0, k-1) \leftarrow (\omega^\beta)^n$
/* Input multiplier for column direction in H2*/
else
H2 $(\alpha, \beta, 0, k-1) \leftarrow$ H2 $(\alpha, \beta, 0, k-2)$;
endif;
H1 $(\alpha, \beta, 0, k) \leftarrow$ H1 $(\alpha, \beta, 0, k) \times$ H2 $(\alpha, \beta, 0, k)$;
Step 3: H2 $(\alpha, \beta, 0, n-1) \leftarrow$ H2 $(\alpha, \beta, 0, n-2)$
/*succeeding values of Fourier vector are generated by multiplying the previous values by powers of $(\omega^\alpha)^n$ and then forwarding to the next processor in row direction in H1 registers */
/* multiplier for column direction are propagated to all the processors of 0th row */

Step 4: for $m = 0$ to $n-1$ do in parallel
begin
for $k = 0$ to $n-2$ do
begin
4.1 do in parallel
H2 (α, β, $k+1$, m) $\leftarrow$ H2 (α, β, k, m);
If ($k = 0$) H2 (α, β, k, m) $\leftarrow$ $b_{nm+\beta}$;
else H2 (α, β, k, m) $\leftarrow$ H2 (α, β, $k-1$, m);
4.2 H1 (α, β, $k+1$, m) $\leftarrow$ H1 (α, β, k, m);
4.3 H1 (α, β, $k+1$, m) $\leftarrow$ H1 (α, β, $k+1$, m) $\times$ H2 (α, β, $k+1$, m);
end
H2 (α, β, $n-1$, m) $\leftarrow$ H2 (α, β, $n-2$, m);
/* Last step to insert the element of input vector into the $n-1$-th row. */
end
/* succeeding values are generated by multiplying the previous values by powers of and $(\omega^{\beta})^n$ and then forwarding to the next processor in column direction */
/* all the b values are copied down to the other rows in the same block */

7.1.2 Multiplication Step

Same as algorithm PLT.

7.1.3 Data Movement Step

Same as algorithm PLT.

7.1.4 Addition Step

$\forall$ α, β and y, $0 \le \alpha, \beta, y \le n-1$ do in parallel
begin
for $i = 0$ to $n-2$ do
H1 (α, β, $i+1$, y) $\leftarrow$ H1 (α, β, $i+1$, y) + H1 (α, β, i, y);
/* P(α, β, $n-1$, y) contains the partial sum of n values */
for $j = n-2$ downto 0 do
H1 (α, β, $n-1$, j) $\leftarrow$ H1 (α, β, $n-1$, j) + H1 (α, β, $n-1$, $j+1$);
/* Summing along the last row in each block, the sum of the n^2 data values of the block is finally brought to the processor $P(\alpha, \beta, n-1, 0)$ */
End

7.1.5 Data Arrangement Step

In this step the output data vector C is moved to the 0th row of all the blocks B $(0, \beta)$, $0 \le \beta \le n-1$.

$\forall$ α and β, $0 \le \alpha, \beta \le n-1$ do in parallel
H1 $(0, \beta, 0, \alpha) \leftarrow$ H1 $(\alpha, \beta, n-1, 0)$; // single step vertical data movement along inter-block link

7.2 Time Complexity of the Algorithm PDFT

Initialization steps involve $3n+1$ data movement steps [$(n+2)$ for steps 1, 2 and 3 and $2(n-1)+1$ for step 4] and $2(n-1)$ multiplication steps [$(n-1)$ multiplication along row direction and $(n-1)$ along column direction] to generate the matrix A, though single step of product is required to compute partial components $a_i b_j$ in step B. Steps C, D and E require n, $2(n-1)$ and 1step of data movements respectively. Step D also requires $2(n-1)$ steps of addition. Time complexity of algorithm PDFT is, therefore, $6n$ steps of data movement, $2n-1$ steps of multiplication and $2n-2$ steps of addition, which is $O(n)$.

Comparing the algorithm PDFT with PLT, we see that though the order of time for data movements and addition steps are same for both, multiplication time is dependent on n for the former. The reason for this is, in case of linear transformation the matrix A was raw data which was input through the left boundary of each block. Vector B was given input through the upper boundary of each block. So, block I/O was more and two boundaries were used. In case of DFT the matrix A has a special property and our parallel algorithm exploited that property to generate n^4 data elements of A by giving only n^2 values as input and that too in parallel to the upper left corner of each block of the MM, thereby reducing input time and area.

7.3 Example

To explain the generation of elements of a block $M(\alpha, \beta)$ from a single input at (0,0) processor and explain the time complexity, the data movements and multiplication steps have been shown in this example taking a 4×4 mesh. Figure 6 shows the 0th row generation and Fig. 7 shows the steps for generation of other rows from the 0th row. For each processor there are two local registers H1 and H2. H1s contain elements of A. Initially, register H2s contain the row multipliers for generation of 0th row, then they will contain the column multipliers to generate the other rows, and finally they will contain the elements of vector B. The example shows the element generation of block $M(1, 2)$. Correspondingly, the row element $R_e = a_{1,2} = \omega^{1\times 2}$, row multiplier $M_R = (\omega^1)^4 = \omega^4$, column multiplier $M_C = (\omega^2)^4 = \omega^8$, and the B vector is $(b_j, b_{j+4}, b_{j+8}, b_{j+12}) = (b_2, b_6, b_{10}, b_{14})$. Figure 8 shows the contents of block $M(1, 2)$ after generation of all the elements where $x = \omega^2 = (1-i)/\sqrt{2}$.

Figure 9 shows the product of contents of registers H1 and H2. Here, $p_{i,j} = a_{i,j} \times b_j$. As the elements $p_{i,j}$'s are now residing in different blocks, they are to be brought together in proximity for adding them to compute the elements of vector C. Figure 10 shows the contents of the meshes after n-steps of horizontal data movements along the inter-block links of MM network. In both Figs. 9 and 10 only 2nd row of meshes and their horizontal inter-block links are shown.

The contents of each mesh now can be added in $2(n-1)$ steps of data movements and additions within the mesh to get the components of vector C in $(n-1, 0)$

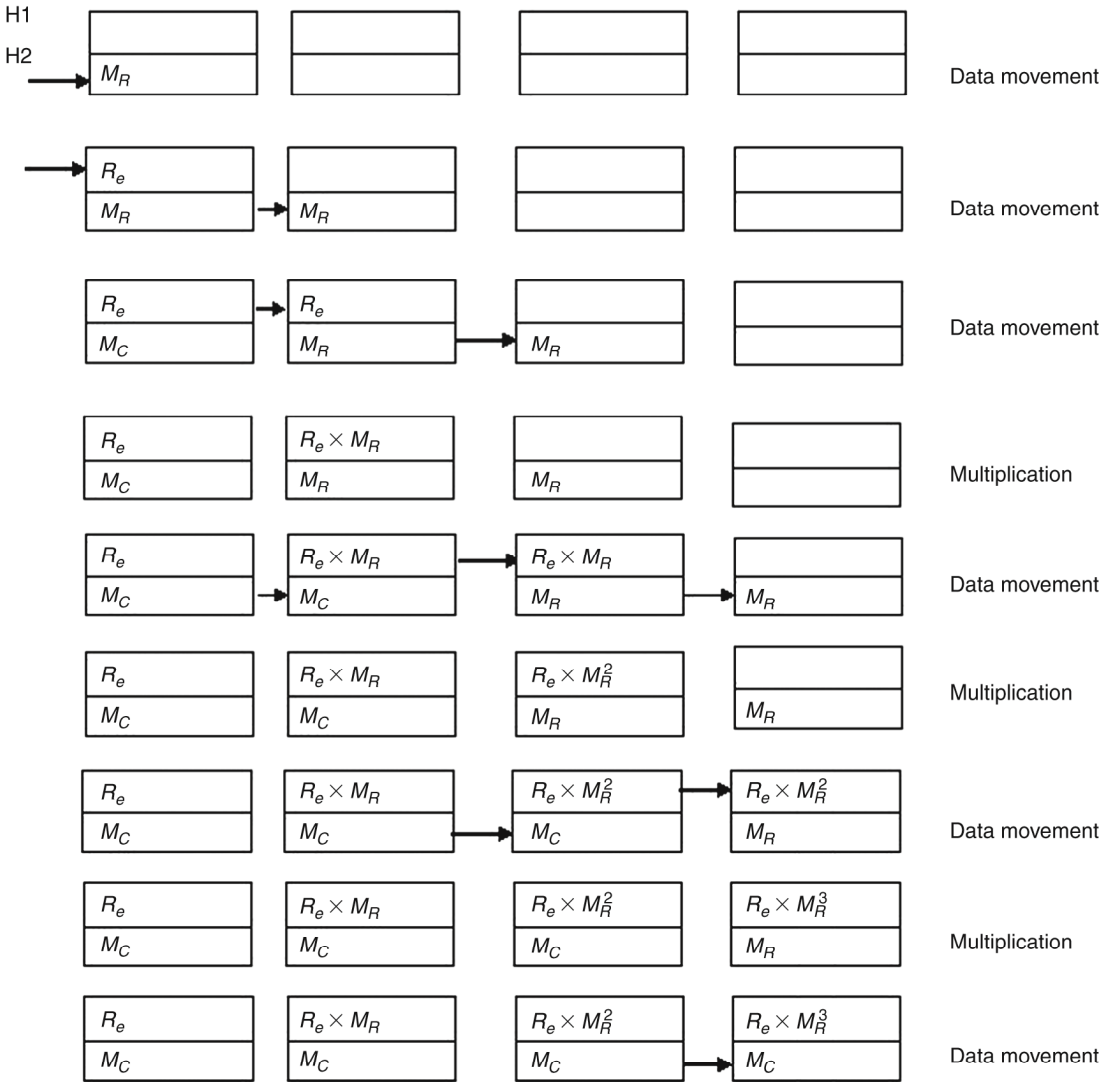


Figure 6. Generation of 0th row using row element and row multiplier of a block.

processor of all the blocks. One single step of data movement along the vertical inter-block link will bring the resultant C vector in the upper boundary of the MM network.

Now, it is evident from Fig. 6 that the number of multiplication steps is 3 and number of data movements steps is $3+3=6$, for generation of the first row, whereas, Fig. 7 shows that three steps of multiplication and seven data movement steps are required for the generation of other rows. Generation of $p_{i,j}$'s in Fig. 9 will involve single step of multiplication. Four steps of horizontal data movements are required to get all the elements as shown in Fig. 10. Six data movement steps and additions are required to get the vector C. Finally, single step of data movement along vertical inter-block link bring the C vector in the upper boundary of the MM network.

7.4 Parallel DFT Computation of Higher Order Matrices

The approach is same as that of parallel implementation of linear transformation of higher order matrices as discussed in Section 6.4. For $k=2^r$, for positive integer value of r, k^2 sub-matrices of A can be generated from the values generated in the algorithm PDFT for each block of MM by properly transforming according to (6).

8. Comparative Study of Time Complexities of Parallel DFT in MM and Other Architectures

The time complexities of parallel DFT computation in various architectures have been tabulated in Table 2.

The comparative study shows that our algorithm performs better than 2D mesh. For the purpose of comparison, we are considering same mesh size $(n \times n)$ and same total number of nodes (n^4) for both MM and multi-dimensional mesh. The diameters of MM and 4D mesh are $2n$ and $4(n-1)$ respectively, and degree of each node in MM is uniformly 4, whereas the node degree of each internal node is 8 for 4D mesh. If smaller meshes are used to construct the multi-dimensional mesh, the node degree will increase. Moreover, the time complexity of proposed DFT computation in MM outperforms multi-dimensional mesh for practical sizes of meshes, that is, mesh sizes less than 2048×2048.

In hypercube, better time complexity is achieved through its high node degree and number of links with increasing dimensions.

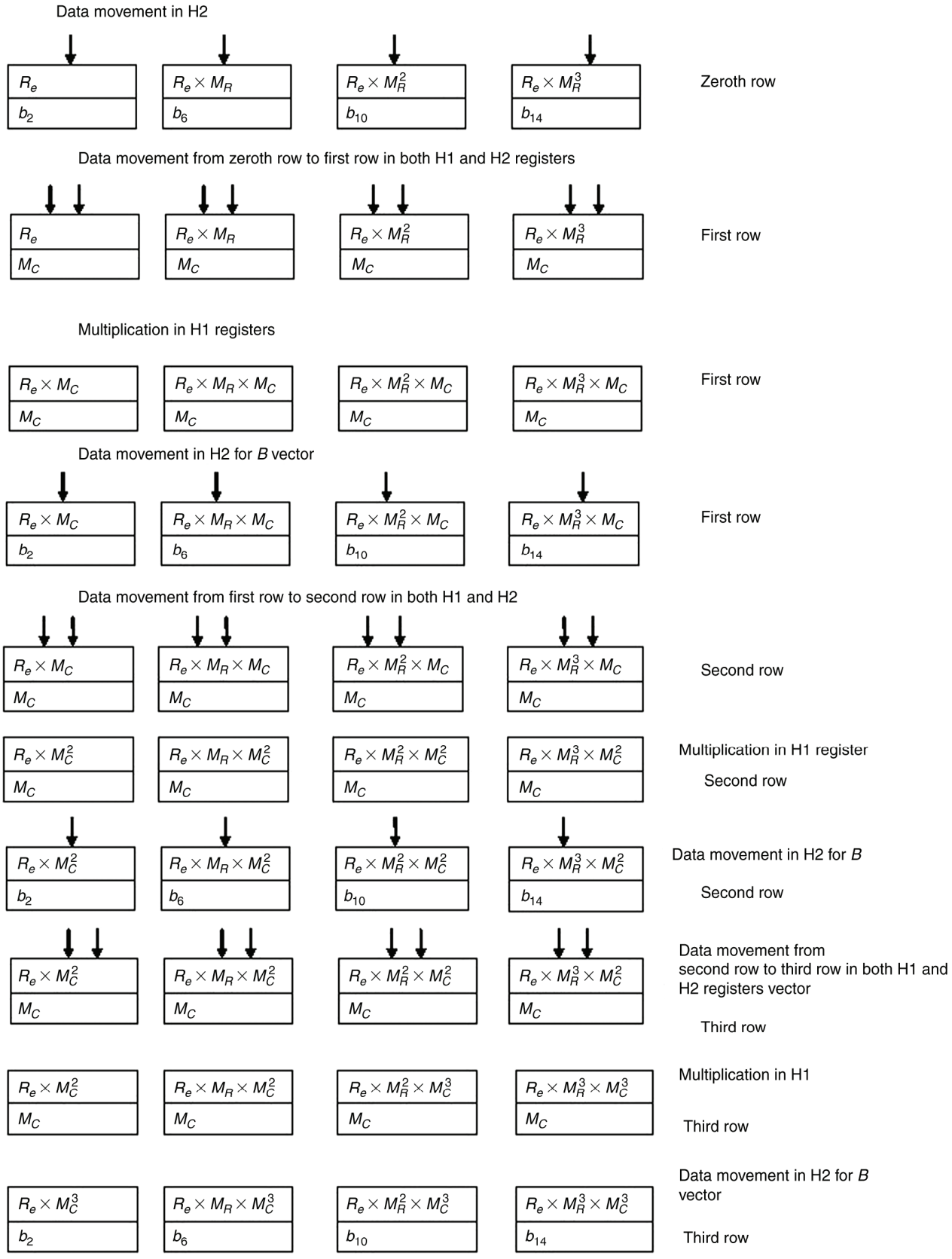


Figure 7. Input of vector B and generation of other rows using column multiplier.

Fast implementation of DFT algorithm on architecture with constant degree processors is also an area of recent research interest. A number of parallel algorithms developed for DFT/FFT in different architectures can be found in the literature [5]–[9], [11], [14], [30]–[32], [34], [35], [36]–[38]. Parallel implementation of Fourier transform has been discussed and implemented in mesh [30]–[32], binary tree [19] and star graph [34].

x	$-xi$	$-x$	xi
$-x$	xi	x	$-xi$
x	$-xi$	$-x$	xi
$-x$	xi	x	$-xi$

Figure 8. Contents of H1 registers of block $M(1, 2)$ where $x = \omega^2 = (1-i)/\sqrt{2}$ and $33 = \cos(\pi/8) - i \times \sin(\pi/8)$.

Bliss and Julien [31] have suggested four architectures using the fast two-dimensional (2D) algorithm for DFT that achieve the maximum throughput per chip area. The first two use N-element 2D meshes requiring $N + 2N^{3/2}$ multiplications and $1 + 2\sqrt{N}$ periods per DFT. The third and fourth architectures both use pipelined DFT blocks of length $\sqrt{N}$ connected by a pipelined matrix transpose. The third uses systolic 2D meshes for the short DFTs with period $\sqrt{N}$. Gertner and Rofheart [35] have proposed a parallel algorithm for the 2D DFT computation which eliminates inter-processor communications and uses only $O(N)$ processors. Shousheng and Torkelson [32] have shown that a simple planar 2D systolic array having $N = M^2$ processing elements can be used to compute DFT of size N in $2M + 1$ steps of pipelined operations, achieving the area-time complexity $AT^2 = O(N^2 \log^3 N)$. They have used an extension of the common factor algorithm. This architecture has also very good expansibility that a 2^t N size DFT transform can be computed on 2^t nearest-neighbour connected N-size arrays with reloaded twiddle factors, which makes it more suitable for VLSI implementation of DFT transform in various practical sizes.

Zhang and Yuns [33] have shown that using n-dimensional mesh parallel DFT computation can be achieved in $O(\log^2 n)$ time using $\mathrm{N} = n^d$ processors.

The interconnection pattern between the processors in multi-dimensional mesh and MM is entirely different although the basic building blocks are 2D meshes in both the cases. Considering $n \times n$ 2D mesh as the basic building

Table 2
Time Complexities of Parallel DFT Computation in Various Architectures

Architecture	Number of Nodes	Length of the Vector to be Transformed	Time Complexity
2D mesh [30], [31], [32]	N	N	$N + 2N^{3/2}$
Multi-dimentional mesh [33]	N	N	$O(\log^2 N)$
Hypercube [34]	N (=n!)	N	$O(\log N)$
Binary tree [19]	$O(N)$	N	$O(N \log N)$
MM [3]	N	$N^{0.4}$	$O(N^{1/4})$
Star graph [34]	n!	n!	$O(n^2)$
MM (Proposed)	N	$N^{1/2}$	$O(N^{1/4})$

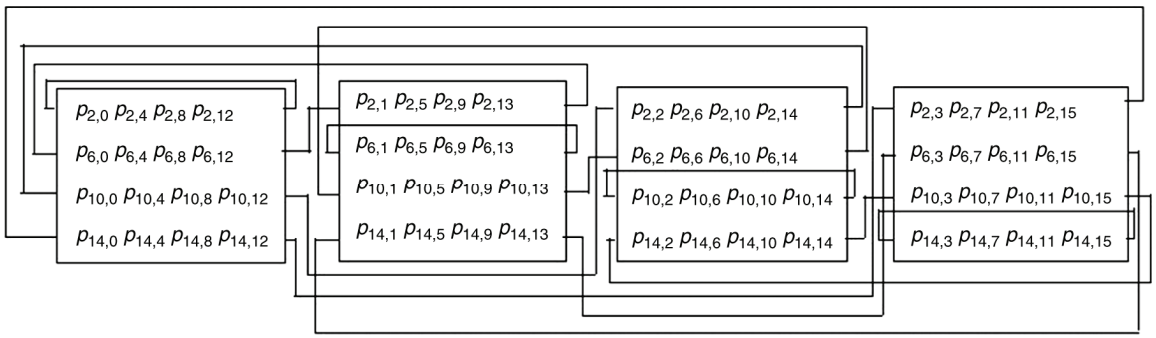


Figure 9. The product $p_{i,j} = a_{i,j} \times b_j$ in the algorithm PDFT computed by multiplying the contents of H1 registers by the content of corresponding H2 registers in second row of meshes in MM for $n = 4$.

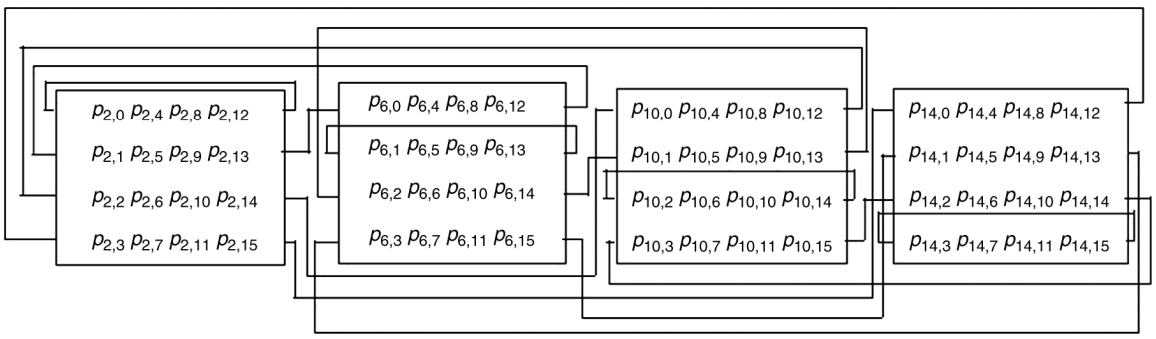


Figure 10. The partial product in the algorithm PDFT after n data movement steps along the horizontal inter-block links in MM for $n = 4$.

Table 3
Computation Time for DFT in Mesh, PLT and PDFT in MM

Total Number of Processors $= N = n^4$. Vector Size to be Transformed $= n^2$									
Algorithm	Fourier Transform in Mesh of Size $n^2 \times n^2$			PLT in MM Where Size of Each Block is $n \times n$			PDFT in MM Where Size of Each Block is $n \times n$		
	Value of n^2			Value of n			Value of n		
	16	64	256	4	8	16	4	8	16
Total addition steps	15	63	255	6	14	30	6	14	30
Total multiplication steps	1	1	1	1	1	1	7	15	31
Total data movement steps	31	127	511	19	39	79	24	48	96
Time complexity	$O(n^2)$			$O(n)$			$O(n)$		

blocks and total number of elements n^4, the four-dimensional mesh architecture has diameter $4(n-1)$ and degree of each internal node is 8 whereas for MM diameter is $2n$ and degree of each node is 4.

It is evident from the table that the time complexity for parallel DFT computation is minimum for MM network.

9. Experimental Results

The PLT described in Section 6 and PDFT described in Section 7 using MM have been simulated through C programs and same input which is transformed using these programs has been set as input to a parallel DFT in mesh architecture. In all the cases the output transformed vector is same. The results of the simulation programs using different sizes of the input vector are shown in Table 3. MM used is also of different sizes. As expected the calculated time complexities matched with that of the theoretical complexities calculated from the algorithm described in Sections 6 and 7. The multiplication steps are high for PDFT as the algorithm generates n^2 (n^2-1) elements of Fourier matrix, whereas in PLT all the n^4 elements were input through the left boundaries of the meshes. For mesh of size $n^2 \times n^2$, addition and data movement steps are n^2-1 and $2n^2-1$, respectively. These numbers of steps are high compared to those of PLT and PDFT in MM as the diameter of mesh is high compared to the diameter of MM.

10. Conclusion and Future Work

A parallel algorithm for linear transformation of the form $AB = C$, where A is a square matrix of order n^2 and B, C are vectors of length n^2, is proposed and implemented in an architecture called MM of n^4 nodes of degree 4 and having diameter $2n$. The time complexity of the algorithm is $O(n)$ (as opposed to $O(n^2)$ in case of a simple mesh having the same number of processors) and the AT^2 value is $O(n^6)$. Then the algorithm is modified for DFT and is implemented in the MM network in $O(n)$ time. For the DFT, instead of the whole matrix A, the input is restricted to only n^2 values of the sub-matrix A_{00} and the rest are calculated in the algorithm itself by proper manipulation (using symmetry and redundancies in the definition of the Fourier matrix A) of these n^2 values which takes $O(n)$ multiplication steps.

As a future work, we may use the generalized MM network for DFT where we may relax the restriction that the number of processors should be n^4, for some n. The difference between the number of processors of two successive MM networks (*i.e.*, for two consecutive values of n) is $(n+1)^4 - n^4$, which increases as $O(n^3)$. This difference, can, however, be reduced if, instead of taking $n \times n$ meshes as the constituent blocks, we take $m \times n$ meshes for any m, $n \geq 3$, arranging mn number of such meshes in the form of an $n \times m$ matrix. The inter-block links can be defined in the same manner as in Section 4. The diameter of such a network can be found to be $m+n$, whereas the total number of processors is m^2n^2. The algorithm for DFT can be suitably restructured to fit into the generalized version of the MM network.

Acknowledgement

The authors are thankful to the reviewers for their valuable comments and suggestions. They also express their sincere gratitude to the faculty members of the Department of Engineering & Technological Studies, University of Kalyani, West Bengal, India, where the work has been carried out with the financial support of PURSE scheme, DST.

References

[1] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to algorithms* (Cambridge, MA: MIT Press, 1989).

[2] S.G. Akl, *The design and analysis of parallel algorithms* (New York, NY: Prentice-Hall, 1989).

[3] D. Das, M. De, and B.P. Sinha, A new network topology with multiple meshes, *IEEE Transactions on Computers*, *48*(5), 1999, 536–551.

[4] B.P. Sinha, Multi-mesh an efficient topology for parallel processing, *Proc. 9th Int. Parallel Processing Symp.* (*IPPS*), Santa Barbara, CA, 1995, 17–21.

[5] A. Dardalis, D. Andreas, and V.K. Prasanna, Fast parallel implementation of DFT using configurable devices, *Proc. Int. Workshop on Field Programmable Logic and Applications*, London, 1–3 September, 1997, 314–323.

[6] M.V. Aliev, A.M. Belov, A.V. Ershov, and M.A. Chicheva, Parallel algorithms for a hyper complex discrete Fourier transform, *Pattern Recognition and Image Analysis*, *15*(1), 2005, 110–112.

[7] Z.C. Xiang, H.G. Qiang, and H.M. He, Some new parallel fast Fourier transform algorithms, *Proc. Sixth Int. Conf. on Parallel and Distributed Computing Application and Technology* (*PDCAT*), 5–8 December, Dalian, China, 2005, 624–628.

[8] R.A. Al Na'mneh, W.D. Pan, and S.M. Yoo, Parallel implementation of 1-D FFT without inter process communication, *International Journal of Computers and Application*, *29*(2), 2007, 180–186.

[9] S. Gurevich, R. Hadani, and N. Sochen, The finite harmonic oscillator and its applications to sequences, communication and radar, *IEEE Transactions on Information Theory*, *54*(9), 2008, 4239–4253.

[10] J.M. Cooley and J.W. Tukey, An algorithm for the machine computation of the complex Fourier series, *Mathematics of Computation*, *19*, 1965, 297–301.

[11] B.P. Sinha and A. Mukherjee, Parallel sorting algorithm using multiway merge and its implementation on a multi-mesh network, *Journal of Parallel and Distributed Computing*, *60*, 2000, 891–960.

[12] B.P. Kundu, M. De, and B.P. Sinha, Wormhole routing for complete exchange in multi-mesh, *Proc. 4th Int. Conf. on High Performance Computing* (*HIPC*), Bangalore, India, 1997, 432–437.

[13] M. De, D. Das, and B.P. Sinha, An efficient sorting algorithm on the multi-mesh network, *IEEE Transactions on Computers*, *46*(10), 1997, 1132–1137.

[14] J.H. Reif and A. Tyagi, Efficient parallel algorithms for optical computing with discrete Fourier transform primitive, *Applied Optics*, *36*(20), 1997, 7327–7340.

[15] N. Afroz, S. Bandyopadhyay, R. Islam, and B.P. Sinha, On the implementation of links in multi-mesh network using WDM optical networks, *Proc. 7th Int. Workshop on Distributed Computing*, Kharagpur, India, December 27–30, 2005 (*LNCS 3741*), 183–188.

[16] A. Sen, S. Bandyopadhyay, and B.P. Sinha, A new architecture and a new metric for lightwave networks, *Journal of Lightwave Technology*, *19*(7), 2001, 913–925.

[17] S. Murthy and A. Sen, A peer-to-peer network based on multi-mesh architecture, *Global Telecommunications Conf.*, 7, 1–5 December, San Francisco, USA, 2003, 3840–3844.

[18] I.D. Scherson and S. Sen, Parallel sorting algorithm in two-dimensional VLSI models, *IEEE Transactions on Computers*, *38*(2), 1989, 238–249.

[19] J. Ja Ja, *An introduction to parallel algorithms* (MA: Addition-Wesley, 1992).

[20] A. Gupta, Mesh based algorithm for finite exponential function, *Proc. 2nd Int. Conf. on Advanced Computing and Communication Technologies*, Rothak, Haryana, India, 2012, 328–330.

[21] Q. Jianxian, A time-space optimal parallel sorting on a hypercube, *University Journal of Natural Sciences*, *1*(3/4), 1996, 465–469.

[22] E. Dekkel, D. Nassimi, and S. Sahni, Parallel matrix and graph algorithms, *SIAM Journal on Computing*, *10*(10), 1981, 657–673.

[23] C.P. Katti and R. Kumari, A new parallel algorithm for Lagrange interpolation on a hypercube, *Computers & Mathematics with Applications*, *51*(6–7), 2006, 1057–1064.

[24] E.T. Leighton, *Introduction on parallel algorithms and architectures: Array, trees and hypercubes* (San Mateo, CA: Morgan Kaufmann, 1992).

[25] D.K. Mallick and P.K. Jana, Parallel prefix on mesh of trees and OTIS mesh of trees, http://nguyendangbinh.org/proceedings/IPCV08/Papers/PDP4269.pdf.

[26] P.K. Jana and B.P. Sinha, Fast parallel algorithm for polynomial interpolation, *29*(4), 1995, 85–92.

[27] C.D. Thompson and H.T. Kung, Sorting on a mesh connected parallel computer, *Communications of the ACM*, *20*(4), 1977, 263–271.

[28] A. Aggarwal, A comparative study of Xtree, pyramid and related machines, *Proc. 25th Annual Symp. on Foundation of Computer Science*, October, 1984, IEEE, New York, NY, 89–99.

[29] A. Grama, A. Gupta, G. Karypis, and V. Kumar, *Introduction to parallel Computing*, 2nd ed. (Boston, MA: Addison Wesley, 2003).

[30] J.P. Strong, The Fourier transform on mesh connected processing arrays such as massively parallel processors, *Proc. 1985 IEEE Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, CAPAIDM, Miami Beach, Florida, 1985, 190–196.

[31] W.G. Bliss and A.F. Julien, Efficient and reliable VLSI algorithms and architectures for the discrete Fourier transform, *Proc. Int. Conf. on Acoustics Speech and Signal Processing*, *2*(3–6), 1990, 901–904.

[32] H. Shousheng and M. Torkelson, A systolic array implementation of common factor algorithm to compute DFT, *Int. Symp. on Parallel Architectures, Algorithms and Networks* (*ISPAN*), December 14–16, Kanazawa, Japan, 1994, 374–381.

[33] C.N. Zhang and D.Y.Y. Yuns, Multidimensional systolic networks for discrete Fourier transform, *Proc. ISCA '84 11th Annual Int. Symp. on Computer Architecture*, Ann Arbor, MI, June, 1984, 215–222.

[34] P. Fragopoulou and S.G. Akl, A parallel algorithm for computing Fourier transform on the star graph, *Transaction on Parallel and Distributed Systems*, *5*(5), 1994, 525–531.

[35] I. Gertner and M. Rofheart, A parallel algorithm for 2D DFT computation with no interprocess communication, *IEEE Transaction on Parallel and Distributed Systems*, *1*(3), 1990, 377–382.

[36] N. Rakesh, Analysis of multi-sort algorithm on multi-mesh of trees (MMT) architecture, *The Journal of Supercomputing*, *57*(3), September 2011, 276–313.

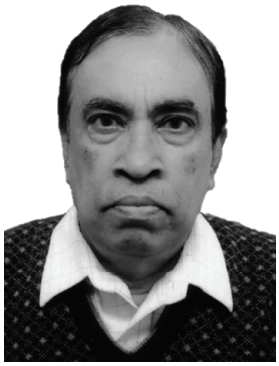
[37] W. Bliss and A.W. Julien, Efficient and reliable VLSI algorithms and architectures for the discrete Fourier transform, *Int. Conf. on Acoustics, Speech and Signal Processing* (ICASSP), April 3–6, Albuquerque, NM, 1990, 901–904.

[38] F. Qureshi and O. Gustafsson, Generation of all radix-2 fast Fourier transform algorithms using binary trees, *Proc. 20th European Conf. on Circuit Theory and Design* (ECCTD), August 29–31, Linkoping, Sweden, 2011, 667–680.

Biographies

Somen De did his Ph.D. from the Department of Physics, University of Kalyani. Presently he is assistant professor in the Department of Physics, Bijoy Krishna Girls' College, Howrah, India. He has published more than 10 research papers in international journals.

Amit Datta received his B.Tech. and M.Tech. degrees in I.T. from University of Kalyani in 2003 and University of Calcutta in 2005, respectively. Presently, he is a Ph.D. research fellow at Department of Engineering & Technological Studies under University of Kalyani. His research interest is on parallel computing & architectures and fault-tolerant computing. He is also a member of IEEE.



Asit B. Bhattacharya did his Ph.D. from the Department of Physics, University of Calcutta and Post-doc from the Massachusetts Institute of Technology, MA, USA. Presently he is a professor in the Department of Physics, Kalyani University and a fellow of the Institute of Electronics and Telecommunication Engineers. He worked in close collaboration with the leading laboratories like Lincoln Laboratory, Milestone Hill Observatory, Earth, Atmosphere and Planetary Sciences of MIT, MA, USA. He has published more than 200 research papers in international journals and guided many Ph.D. students. He is a reviewer of many scientific journals and his major field of work has been in antenna, astronomical radio spectrograph, solar-terrestrial physics and remote sensing. He is the author of 14 text books on science and engineering topics which includes astronomy and astrophysics (Infinity Science Press, Hingham, MA, USA), "Search for Extraterrestrial Intelligence", "Particle Physics as a building block of the Universe" and "The Ionosphere and its Transient Variations" (Lap Lambert Academic Publishing, Germany).

Mallika De received her B.Sc. degree in Physics from Calcutta University in 1973 and the M.Sc. degree in Applied Mathematics from Jadavpur University in 1976. She received the Advanced Diploma in Computer Science and M.Tech. in Computer Science in the year 1980 and 1985, respectively, from Indian Statistical Institute, Calcutta. Her Ph.D. degree in Engineering was awarded in the year 1997 from Jadavpur University. She is currently a senior faculty of the Department of Engineering & Technological Studies at University of Kalyani, where she is serving for last 28 years as faculty. Her research interest includes parallel algorithms & architectures, fault-tolerant computing, image processing, soft computing and quantum cellular automata. She has authored/coauthored 35 refereed journal articles and more than 30 conference papers. She has worked as paper reviewer for few international conferences such as advanced computing & communication, high performance computing and Asian test symposium.